ﺗﻤﺮﯾﻦ دوم

**ﻣﺤﻤﺪﻋﺮﻓﺎن ﻫﻮﻣﺎﻧﻔﺮ**

ﺑﺨﺶ اول:

# Business Requirement

١(داﺷﺒﻮرد ﻣﺪﯾﺮﯾﺖ ﮐﺎرﺑﺮان:

ﺻﺎﺣﺐ ﺳﯿﺴﺘﻢ داﺷﺒﻮردی ﺟﻬﺖ ﻣﺪﯾﺮﯾﺖ ﮐﺎرﺑﺮاﻧﯽ ﮐﻪ از ﺳﯿﺴﺘﻢ اﺳﺘﻔﺎده ﻣﯿﮑﻨﻨﺪ را داﺷﺘﻪ ﺑﺎﺷﺪ ﺗﺎ ﺗﻮاﻧﺎﯾﯽ اﺿﺎﻓﻪ

ﮐﺮدن ﮐﺎرﺑﺮ ﺟﺪﯾﺪ و درﺻﻮرت ﻧﯿﺎز ﻏﯿﺮﻓﻌﺎل ﮐﺮدن ﮐﺎرﺑﺮان ﻓﻌﻠﯽ ﺗﻮﺳﻂ ادﻣﯿﻦ اﺻﻠﯽ ﺳﯿﺴﺘﻢ وﺟﻮد داﺷﺘﻪ ﺑﺎﺷﻨﺪ.

در ﺳﯿﺴﺘﻢ ﻣﺎ وﺟﻮد ﭘﻨﻞ ادﻣﯿﻦ اﯾﻦ اﻣﮑﺎن را ﻓﺮاﻫﻢ ﻣﯿﮑﻨﺪ ﮐﻪ ﮐﺎرﺑﺮاﻧﯽ ﮐﻪ از ﺳﯿﺴﺘﻢ اﺳﺘﻔﺎده ﻣﯿﮑﻨﻨﺪ ﻣﺪﯾﺮﯾﺖ

ﺷﻮﻧﺪ.

٢( ﺗﻌﻤﯿﺮ و ﻧﮕﻬﺪاری ﺳﯿﺴﺘﻢ ﻣﻘﺮون ﺑﻪ ﺻﺮﻓﻪ را از ﻃﺮﯾﻖ اﺗﻮﻣﺎﺳﯿﻮن ﻓﻌﺎل ﺑﺎﺷﺪ ﺗﺎ ﺑﺎ اﺟﺮا و ﻧﺼﺐ ﺧﻮدﮐﺎر و رﻓﻊ

اﺗﻮﻣﺎﺗﯿﮏ ﻫﺸﺪار ﻫﺎ ﺳﺮﻋﺖ ﻋﻤﻞ اﻓﺰاﯾﺶ و ﻫﺰﯾﻨﻪ ﻫﺎ ﮐﺎﻫﺶ ﭘﯿﺪا ﮐﻨﺪ.

در ﺳﯿﺴﺘﻢ ﻣﺎ اﺑﺰار اﺗﻮﻣﺎﺳﯿﻮن ﺑﺮای ﻧﺼﺐ و راه اﻧﺪازی ﺳﯿﺴﺘﻢ ﺑﻪ ﻋﻤﻞ آورده ﺷﺪه ﮐﻪ ﻣﻮﺟﺐ ﺷﺪه ﻧﺼﺐ zabbix و ﻣﻮارد ﻣﻮردﻧﯿﺎز ﺑﺮروی ﺳﯿﺴﺘﻢ ﻫﺎ راﺣﺖ ﺗﺮ ﺷﻮد ﻫﻤﭽﻨﯿﻦ ﺑﺎ ﺑﻪ ﮐﺎر ﺑﺮدن ﻋﻤﻞ ﻫﺎﯾﯽ از ﺟﻤﻠﻪ raid ﺑﻨﺪی ﺣﺎﻓﻈﻪ ، downtime ﺳﯿﺴﺘﻢ درﺻﻮرت رخ دادن ﻣﻮاردی ﻫﻤﭽﻮن از دﺳﺖ دادن ﻧﺎﮔﻬﺎﻧﯽ دﯾﺴﮏ ﮐﺎﻫﺶ

ﯾﺎﺑﺪ



# Customer Requirement

١( ﻣﺸﺘﺮی ﻧﯿﺎزﻣﻨﺪ راﺑﻂ ﮐﺎرﺑﺮی اﺳﺎن ﺑﺮای ﻣﺸﺎﻫﺪه وﺿﻌﯿﺖ ﺳﯿﺴﺘﻢ ﺧﻮد ﻫﺴﺖ ﺳﯿﺴﺘﻢ دارای داﺷﺒﻮرد راﺣﺖ ﺑﺎ دﺳﺘﺮﺳﯽ ﻣﺸﺨﺺ و ﻫﻤﭽﻨﯿﻦ دارای ﻣﻮاردی ﻫﻤﭽﻮن ﻧﺸﺎن دادن داده ﻫﺎ ﺑﻪ

ﺻﻮرت ﻧﻤﻮداری ﺟﻬﺖ درک راﺣﺖ ﺗﺮ ﮐﺎرﺑﺮان ﻫﺴﺖ

٢( ﭘﺸﺘﯿﺒﺎﻧﯽ از اﻧﻮاع ﻣﺨﺘﻠﻒ ﮐﺎرﺑﺮان ﺗﺎ ادﻣﯿﻦ ﻫﺎ و ﮐﺎرﺑﺮان ﻋﺎدی و ... ﺑﺘﻮاﻧﻨﺪ از ﺳﯿﺴﺘﻢ اﺳﺘﻔﺎده ﮐﻨﻨﺪ در ﺳﯿﺴﺘﻢ ﻣﺎ دو ﻧﻘﺶ ﻣﺘﻔﺎوت ادﻣﯿﻦ ﺳﯿﺴﺘﻢ و ﮐﺎرﺑﺮ زﯾﺮﻣﺠﻤﻮﻋﻪ ادﻣﯿﻦ ﺑﺎ ﺳﻄﺢ دﺳﺘﺮﺳﯽ ﻫﺎی ﻣﺨﺘﻠﻒ وﺟﻮد

دارد ﺗﺎ درﺻﻮرت ﭘﯿﭽﯿﺪه ﺗﺮ ﺑﻮدن ﺷﺮﮐﺖ ﺑﺘﻮان دﺳﺘﺮﺳﯽ اﻓﺮاد ﻣﺘﻔﺎوت را ﻣﺤﺪود و ﻫﻤﭽﻨﯿﻦ ﺑﺎ ﭼﻨﺪاﮐﺎﻧﺖ

ﻣﺘﻔﺎوت ﺑﺮروی ﺳﯿﺴﺘﻢ وارد ﺷﺪ و ﻧﻈﺎرت ﮐﺮد



# Architectural Requirement

١( ﺳﯿﺴﺘﻢ ﺑﺎﯾﺪ دارای ﻣﻌﻤﺎری ﻣﯿﮑﺮوﺳﺮوﯾﺲ ﺑﺎﺷﺪ ﺗﺎ ﻣﻘﯿﺎس ﭘﺬﯾﺮی و اﺟﺰا ﺑﻪ ﺻﻮرت ﻣﺴﺘﻘﻞ ﮐﺎرﮐﻨﻨﺪ

در ﺳﯿﺴﺘﻢ ﻫﺴﺘﻪ اﺻﻠﯽ ﺑﺮای درﯾﺎﻓﺖ اﻃﻼﻋﺎت ﺳﯿﺴﺘﻢ از ﺑﮑﻨﺪ و ﻓﺮاﻧﺖ اﻧﺪ ﮐﺎﻣﻼ ﺟﺪا ﺷﺪه اﻧﺪ ﺗﺎ درﺻﻮرت ﻧﯿﺎز ﺑﺮای اﯾﺠﺎد ﺗﻐﯿﯿﺮ در ﻧﺤﻮه درﯾﺎﻓﺖ اﻃﻼﻋﺎت ﻣﺎﻧﯿﺘﻮرﯾﻨﮓ ﺳﯿﺴﺘﻢ ﻧﯿﺎزی ﺑﻪ ﺗﻐﯿﯿﺮ ﻣﻨﻄﻖ ﺑﮑﻨﺪ ﻧﺒﺎﺷﺪ و ﻣﻨﻄﻖ ﺑﮑﻨﺪ

ﺑﺘﻮاﻧﻨﺪ ﺑﺎ ورژن ﻫﺎی ﻣﺨﺘﻠﻒ ﻫﺴﺘﻪ ﻣﺎ ﮐﺎرﮐﻨﺪ

٢( ﺳﯿﺴﺘﻢ ﺑﺎﯾﺪ ﯾﮏ ﻣﺪل ارﺗﺒﺎﻃﯽ ﻣﺒﺘﻨﯽ ﺑﻪ ﺻﻒ ﺑﺮای ارﺳﺎل ﭘﯿﺎم ﻫﺎ ﺑﺮای ارﺳﺎل ﻫﻤﺰﻣﺎن داﺷﺘﻪ ﺑﺎﺷﺪ ﺳﯿﺴﺘﻢ ﺑﺎ اﺳﺘﻔﺎده از ﻣﻮارد ارﺳﺎﻟﯽ ﻫﻤﭽﻮن broker message ﻫﺎ اﯾﻦ اﺧﺘﯿﺎر را ﺑﻪ ﻣﺎ ﻣﯿﺪﻫﺪ ﺗﺎ ﺑﺘﻮاﻧﯿﻢ ﭘﯿﺎم ﻫﺎرا

ﺑﻪ ﺻﻮرت ﻣﺪﯾﺮﯾﺖ ﺷﺪه ﺑﻪ ﮐﺎرﺑﺮان ارﺳﺎل ﮐﻨﯿﻢ ﺗﺎ زﻣﺎن اﻃﻼع رﺳﺎﻧﯽ ﺑﻪ ﮐﺎرﺑﺮان ﺑﻪ ﺣﺪاﻗﻞ ﺑﺮﺳﻨﺪ



# Structural Requirement

١( داده ﻫﺎی ﮐﺎرﺑﺮان ﺑﺎﯾﺪ ﺑﻪ ﺻﻮرت اﯾﻤﻨﯽ ذﺧﯿﺮه ﺳﺎزی ﺷﻮد و ﺑﺎ ﺗﻮﺟﻪ ﺑﻪ ﻧﻘﺶ ﻫﺎی ﻫﺮﮐﺎرﺑﺮ و دﺳﺘﺮﺳﯽ

ﻫﺮﮐﺎرﺑﺮ داده ﻫﺎ ﺑﻪ وی ﻧﻤﺎﯾﺶ داده ﺷﻮد

درﺳﯿﺴﺘﻢ ﻣﺎ ﺑﻪ دﻟﯿﻞ وﺟﻮد ﺟﺪاول دﯾﺘﺎﺑﯿﺲ اﻃﻼﻋﺎت ﮐﺎرﺑﺮ ﻫﻤﺮاه ﺑﺎ ﻧﻘﺶ و دﺳﺘﺮﺳﯽ ﻫﺎی آن ﻫﺎ ذﺧﯿﺮه ﺷﺪه و

اﻃﻼﻋﺎت ﻫﺎ ﺑﺎ ﺗﻮﺟﻪ ﺑﻪ دﺳﺘﺮﺳﯽ ﻫﺎی ﮐﺎرﺑﺮ ﺑﻪ وی ﻧﻤﺎﯾﺶ داده ﻣﯿﺸﻮد ٢( ﺳﺎزﻣﺎﻧﺪﻫﯽ داده ﻫﺎ ﺑﻪ ﺻﻮرت ﺳﻠﺴﻪ ﻣﺮاﺗﺒﯽ ﺑﺮای ﮐﺎرﺑﺮان و ﮐﺎرﺑﺮان زﯾﺮﻣﺠﻤﻮﻋﻪ ﻓﺮاﻫﻢ ﺷﻮد

درﺳﯿﺴﺘﻢ ﻣﺎ ﮐﺎرﺑﺮان ﺑﻪ ﺻﻮرت ﺳﻠﺴﻠﻪ ﻣﺮاﺗﺒﯽ ذﺧﯿﺮه ﺷﺪه اﻧﺪ و ﺑﺎ اﺳﺘﻔﺎده از رﻓﺮﻧﺲ ﻫﺎی دروﻧﯽ دﯾﺘﺎﺑﯿﺲ ﻫﺮﮐﺎرﺑﺮ ﺑﻪ ﮐﺎرﺑﺮ ﺑﺎﻻدﺳﺘﯽ ﺧﻮد ﻣﺘﺼﻞ ﻣﯿﺸﻮد و ﺗﻮاﻧﺎﯾﯽ ﺗﺸﺨﯿﺺ ﮐﺎرﺑﺮان و زﯾﺮﻣﺠﻤﻮﻋﻪ ﻫﺎی ﻫﺮﮐﺎرﺑﺮ وﺟﻮد

دارد



# Behavioral Requirement

١( ارﺳﺎل اﻋﻼن درﺻﻮرت وارد ﺷﺪن ﺳﯿﺴﺘﻢ ﺑﻪ ﺣﺎﻟﺖ ﻫﺎی اﺿﻄﺮاری

در ﺳﯿﺴﺘﻢ ﺑﻪ ﺻﻮرت اﺗﻮﻣﺎﺗﯿﮏ درﺻﻮرﺗﯽ ﮐﻪ ﺳﯿﺴﺘﻢ وارد ﺷﺮاﯾﻂ ﺑﺤﺮاﻧﯽ ﺷﻮد ﯾﺎ درآﺳﺘﺎﻧﻪ ﻗﺮارﮔﯿﺮی در اﯾﻦ ﻧﺎﺣﯿﻪ ﻫﺎ ﻗﺮار ﺑﮕﯿﺮد ﺑﻪ ﮐﺎرﺑﺮان ﺑﻪ ﺻﻮرت اﯾﻤﯿﻞ اﻃﻼع رﺳﺎﻧﯽ ﻣﯿﺸﻮد ﺗﺎ ﻫﺮﭼﻪ ﺳﺮﯾﻊ ﺗﺮ ﺑﺮای رﻓﻊ ﻣﺸﮑﻞ اﻗﺪام

ﮐﻨﻨﺪ. ٢( ﺟﻠﻮﮔﯿﺮی از ﺛﺒﺖ ﮐﺮدن اﻃﻼﻋﺎت ﻏﻠﻂ درﭘﺮوﻓﺎﯾﻞ ﮐﺎرﺑﺮان

ﻫﻨﮕﺎم ﺛﺒﺖ اﻃﻼﻋﺎت ﻣﺮﺑﻮط ﺑﻪ ﺳﺮوﯾﺲ zabbix درﭘﺮوﻓﺎﯾﻞ ، اﺑﺘﺪا اﻃﻼﻋﺎت ﺑﺮرﺳﯽ ﻣﯿﺸﻮﻧﺪ ﮐﻪ اﺗﺼﺎل ﺑﻪ zabbix ﺑﺮﻗﺮار ﻫﺴﺖ ﯾﺎ ﺧﯿﺮ و درﺻﻮرت ﻏﻠﻂ ﺑﻮدن اﻃﻼﻋﺎت اﺟﺎزه ذﺧﯿﺮه ﺳﺎزی داده ﻧﻤﯿﺸﻮد



# Functional Requirement

١( ﺳﯿﺴﺘﻢ ﺑﺎﯾﺪ ﮐﺸﻒ و ﭘﯿﮑﺮﺑﻨﺪی اﺗﻮﻣﺎﺗﯿﮏ ﺑﺮای اﺿﺎﻓﻪ ﮐﺮدن ﻣﻮارد ﻓﯿﺰﯾﮑﯽ ﺟﺪﯾﺪ اراﻳﻪ دﻫﺪ ﺳﯿﺴﺘﻢ ﻣﺎ ﺑﻪ ﺻﻮرت اﺗﻮﻣﺎﺗﯿﮏ درﺻﻮرت اﺿﺎﻓﻪ ﺷﺪن دﯾﺴﮏ و ﻫﺎرد و ﮐﺎرت ﺷﺒﮑﻪ ﺟﺪﯾﺪ ﺑﻪ ﺳﯿﺴﺘﻢ اﯾﻦ ﻣﻮارد را

ﺗﺸﺨﯿﺺ و ﻣﺎﻧﯿﺘﻮرﯾﻨﮓ آن ﻫﺎرا آﻏﺎز ﻣﯿﮑﻨﺪ ﺗﺎ درﺻﻮرت اﺿﺎﻓﻪ ﺷﺪن و ﺗﻐﯿﯿﺮ در اﺟﺰا ﻓﯿﺰﯾﮑﯽ ﻧﯿﺎزی ﺑﻪ ﭘﯿﮑﺮﺑﻨﺪی

دﺳﺘﯽ ﻧﺒﺎﺷﺪ ٢( ﺳﯿﺴﺘﻢ ﺑﺎﯾﺪ اﻃﻼﻋﺎت را ﺑﻪ ﺻﻮرت ﻟﺤﻈﻪ ای ﻧﻤﺎﯾﺶ دﻫﺪ

ﺳﯿﺴﺘﻢ ﻣﺎ اﻃﻼﻋﺎت را ﺑﻪ ﺻﻮرت ﻟﺤﻈﻪ ای ﻧﻤﺎﯾﺶ ﻣﯿﺪﻫﺪ ﮐﻪ اﯾﻦ ﺳﺒﺐ ﻣﯿﺸﻮد ﮐﻪ اﻃﻼﻋﺎت ﺳﯿﺴﺘﻢ ﺑﻪ ﺻﻮرت ﻣﺪاوم آﭘﺪﯾﺖ ﺷﻮﻧﺪ و آﺧﺮﯾﻦ اﺗﻔﺎﻗﺎت ﺳﯿﺴﺘﻢ ﻫﺎ ﻧﻤﺎﯾﺶ داده ﺷﻮﻧﺪ ﺗﺎ ﮐﺎرﺑﺮان از آﺧﺮﯾﻦ وﺿﻌﯿﺖ ﺳﯿﺴﺘﻢ ﺧﻮد

ﻣﻄﻠﻊ ﺑﺎﺷﻨﺪ





# Design Requirement

١( در ﻃﺮاﺣﯽ api ﻣﻮﺟﻮد ﺑﺎﯾﺪ از اﺳﺘﺎﻧﺪارد ﻫﺎ ﭘﯿﺮوی ﮐﻨﺪ ﺳﯿﺴﺘﻢ دارای api ﺟﺪا ﺑﺮای اﺳﺘﻔﺎده ﺑﺮﻧﺎﻣﻪ ﻧﻮﯾﺲ ﻫﺎ ﺟﻬﺖ ارﺗﺒﺎط ﻣﺴﺘﻘﯿﻢ ﺑﺎ ﺳﯿﺴﺘﻢ اﺳﺖ و اﯾﻦ api از

ﺑﺎﺷﺪ اﺳﺘﺎﻧﺪارد ﻫﺎی ﻣﻮﺟﻮد REST ﭘﯿﺮوی ﻣﯿﮑﻨﺪ ٢( ﺳﯿﺴﺘﻢ ﺑﺎﯾﺪ ﻃﺮاﺣﯽ ﻣﻨﺎﺳﺒﯽ ﻧﺴﺒﺖ ﺑﻪ ﺳﻄﺢ ﮐﺎرﺑﺮان ﺑﺮﺧﻮردار

درﺳﯿﺴﺘﻢ ﻣﺎ ﺑﺎ ﺑﻪ ﮐﺎر ﮔﯿﺮی ﮐﻨﺘﺮل ﻣﺒﻨﺘﯽ ﺑﺮ ﻧﻘﺶ (RBAC) ﺑﺎ ﺗﻨﻈﯿﻤﺎت دﺳﺘﺮﺳﯽ ﻣﺘﻔﺎوت اﯾﻦ اﻣﮑﺎن را ﻣﯿﺪﻫﺪ ﮐﻪ

دﺳﺘﺮﺳﯽ ﮐﺎرﺑﺮان ﺗﻮﺳﻂ ادﻣﯿﻦ ﻫﺎ ﻗﺎﺑﻞ ﮐﻨﺘﺮل ﺑﺎﺷﺪ



# Derived Requirement

١( ﺑﺎ ﺗﻮﺟﻪ ﺑﻪ ﻣﻌﻤﺎری ﺟﺪا از ﺳﯿﺴﺘﻢ ﺑﺎﯾﺪ ﺷﺮاﯾﻄﯽ ﺑﺮای اﺗﺼﺎل راﺣﺖ اﺟﺰا ﺑﺪﻫﺪ اﺟﺰا ﺳﯿﺴﺘﻢ ﺑﻪ ﺻﻮرت ﺟﺪا ﻗﺮار ﮔﺮﻓﺘﻪ اﻧﺪ و ﻣﮑﺎﻧﯿﺴﻢ ﻣﻨﺎﺳﺐ ﺟﻬﺖ ﻫﻤﺎﻫﻨﮓ ﺳﺎزی اﺟﺰای ﻣﺨﺘﻠﻒ اﯾﺠﺎد ﺷﺪه ﺗﺎ

اﺗﺼﺎﻻت ﺑﻪ ﻫﻤﺪﯾﮕﻪ اﯾﻤﻦ و ﭘﺎﯾﺪار ﺑﺎﺷﻨﺪ ٢( ﺑﺎ ﺗﻮﺟﻪ ﺑﻪ رﯾﻤﻮت ﺑﻮدن داﺷﺒﻮرد ﺑﺎﯾﺪ اﻣﻨﯿﺖ ﺟﻬﺖ ﻟﻮ ﻧﺮﻓﺘﻦ داﺷﺒﻮرد ﮐﺎرﺑﺮان ﺑﺮای اﻓﺮاد ﻏﯿﺮﻣﺠﺎز ﻣﺪﯾﺮﯾﺖ ﺷﻮد

در ﺳﯿﺴﺘﻢ ﺑﺎاﺳﺘﻔﺎده از ﻣﻮاردی ﻫﻤﭽﻮن ssh و token jwt ﻫﺎ اﯾﻦ ﻣﻮرد اراﯾﻪ ﻣﯿﺸﻮد ﮐﻪ اﻃﻼﻋﺎت ﺑﻪ ﺻﻮرت اﻣﻦ ﺑﯿﻦ ﺳﺮور و ﮐﺎﻣﭙﯿﻮﺗﺮ ﺟﺎﺑﺠﺎ ﺷﻮﻧﺪ ﺗﺎ اﻓﺮاد ﻏﯿﺮ ﻣﺠﺎز ﻧﺘﻮاﻧﻨﺪ ﺑﻪ راﺣﺘﯽ ﺑﻪ اﻃﻼﻋﺎت ﮐﺎرﺑﺮان دﺳﺘﺮﺳﯽ داﺷﺘﻪ

ﺑﺎﺷﻨﺪ



# Allocated Requirement

١( ﺳﯿﺴﺘﻢ اﺣﺮاز ﻫﻮﯾﺖ ﺑﺎﯾﺪ ﺗﻮاﻧﺎﯾﯽ ﮐﺎرﮐﺮدن ﻫﻤﺰﻣﺎن ﭼﻨﺪﯾﻦ ﮐﺎرﺑﺮ را داﺷﺘﻪ ﺑﺎﺷﺪ

ﺳﯿﺴﺘﻢ ﺑﺎ اﺳﺘﻔﺎده از روش ﻫﺎی ﻫﻤﺰﻣﺎﻧﯽ و ﺑﻬﯿﻨﻪ ﺳﺎزی ﻣﺘﻔﺎوت اﯾﻦ اﻣﮑﺎن را ﻣﯿﺪﻫﺪ ﮐﻪ ﻫﻤﺰﻣﺎن ﺗﺎ ٠٠٠١

ﮐﺎرﺑﺮ ﺑﺘﻮاﻧﻨﺪ ﺑﺎ ﺣﺪاﮐﺜﺮ زﻣﺎن 500 ﻣﯿﻠﯽ ﺛﺎﻧﯿﻪ ﺑﻪ ﺳﺮور وارد ﺷﻮﻧﺪ

٢( ﺳﯿﺴﺘﻢ درﯾﺎﻓﺖ اﻃﻼﻋﺎت ﺑﺎﯾﺪ اﯾﻦ ﺗﻮاﻧﺎﯾﯽ را داﺷﺘﻪ ﺑﺎﺷﺪ ﮐﻪ اﻃﻼﻋﺎت ﻣﻮﺟﻮد ﺳﯿﺴﺘﻢ را ﺑﻪ ﺳﺮﻋﺖ درﯾﺎﻓﺖ ﮐﻨﺪ ﺑﺎ ﺗﻮﺟﻪ ﺑﻪ ﮐﺎرﮐﺮدن رﯾﻤﻮت ﺳﯿﺴﺘﻢ اﯾﻦ ﺗﻀﻤﯿﻦ را ﻣﯿﺪﻫﺪ ﮐﻪ ﺑﺎ اﺳﺘﻔﺎده از روش ﻫﺎی ﺑﻬﯿﻨﻪ اﻃﻼﻋﺎت ﺳﯿﺴﺘﻢ را

درﮐﻤﺘﺮﯾﻦ زﻣﺎن ﻣﻤﮑﻦ ﺑﯿﻦ ﺳﯿﺴﺘﻢ درﺣﺎل ﻣﺎﻧﯿﺘﻮر ﺷﺪن و ﺳﺮور و ﺳﯿﺴﺘﻢ رﯾﻤﻮت ﻣﺸﺎﻫﺪه اﻧﺘﻘﺎل دﻫﺪ

ﺑﺨﺶ دوم:

### Cohesion )ﻫﻢﭘﯿﻮﺳﺘﮕﯽ(

**ﻫﻢﭘﯿﻮﺳﺘﮕﯽ** ﺑﻪ ﻣﯿﺰان اﻧﺴﺠﺎم و ﻣﺮﺗﺒﻂ ﺑﻮدن اﺟﺰای ﯾﮏ ﻣﺎژول ﯾﺎ ﮐﻼس اﺷﺎره دارد. ﻫﺮﭼﻪ اﺟﺰا وﻇﺎﯾﻒ ﺑﯿﺸﺘﺮی را ﺑﺎ

ﯾﮑﺪﯾﮕﺮ ﻣﺮﺗﺒﻂ و ﻫﻤﺎﻫﻨﮓ اﻧﺠﺎم دﻫﻨﺪ، ﻫﻢﭘﯿﻮﺳﺘﮕﯽ ﺑﯿﺸﺘﺮ اﺳﺖ. اﻧﻮاع ﻫﻢﭘﯿﻮﺳﺘﮕﯽ ﺑﻪ ﺗﺮﺗﯿﺐ از ﺿﻌﯿﻒﺗﺮﯾﻦ ﺗﺎ

ﻗﻮیﺗﺮﯾﻦ ﺷﺎﻣﻞ ﻣﻮارد زﯾﺮ اﺳﺖ:

#### (ﻫﻢزﻣﺎﻧﯽ) Coincidental Cohesion .1

اﺟﺰای ﻣﺎژول ﮐﺎﻣ ﻼ ﺑﯽرﺑﻂ ﻫﺴﺘﻨﺪ و ﺑﻪﻃﻮر ﺗﺼﺎدﻓﯽ ﮐﻨﺎر ﻫﻢ ﻗﺮار ﮔﺮﻓﺘﻪاﻧﺪ.

ﻣﺜﺎل: در ﯾﮏ ﻣﺎژول ﻫﻢ ﻓﺎﯾﻞ ﺧﻮاﻧﺪه ﺷﻮد، ﻫﻢ ﺗﺎرﯾﺦ ﺳﯿﺴﺘﻢ ﭼﮏ ﺷﻮد، و ﻫﻢ ﭘﯿﺎﻣﯽ در ﻻگ ﺛﺒﺖ ﺷﻮد.

#### (ﻣﻨﻄﻘﯽ) Logical Cohesion .2

اﺟﺰا از ﻧﻈﺮ ﻣﻨﻄﻘﯽ ﺑﻪ ﻫﻢ ﻣﺮﺗﺒﻂ ﻫﺴﺘﻨﺪ، اﻣﺎ اﺟﺮای آنﻫﺎ ﻣﺴﺘﻘﻞ از ﻫﻢ اﺳﺖ.

ﻣﺜﺎل: ﻣﺎژوﻟﯽ ﮐﻪ ﭼﻨﺪﯾﻦ ﻋﻤﻠﯿﺎت ﻣﺨﺘﻠﻒ روی ﻓﺎﯾﻞﻫﺎ را ﺑﺴﺘﻪ ﺑﻪ ورودی ﮐﺎرﺑﺮ اﻧﺠﺎم ﻣﯽدﻫﺪ.

#### (زﻣﺎﻧﯽ) Temporal Cohesion .3

اﺟﺰا وﻇﺎﯾﻔﯽ را اﻧﺠﺎم ﻣﯽدﻫﻨﺪ ﮐﻪ ﺑﺎﯾﺪ در ﯾﮏ ﺑﺎزه زﻣﺎﻧﯽ ﺧﺎص اﻧﺠﺎم ﺷﻮﻧﺪ. ﻣﺜﺎل: ﯾﮏ ﻣﺎژول ﮐﻪ در ﻫﻨﮕﺎم ﺷﺮوع ﺑﺮﻧﺎﻣﻪ وﻇﺎﯾﻔﯽ ﻣﺜﻞ ﺑﺎرﮔﺬاری ﺗﻨﻈﯿﻤﺎت، ﭼﮏ ﮐﺮدن ﻧﺴﺨﻪ و اﺗﺼﺎل ﺑﻪ

دﯾﺘﺎﺑﯿﺲ را اﻧﺠﺎم ﻣﯽدﻫﺪ.

#### (روﯾﻪای) Procedural Cohesion .4

اﺟﺰا ﻣﺮاﺣﻞ ﻣﺨﺘﻠﻒ ﯾﮏ ﻓﺮآﯾﻨﺪ را اﺟﺮا ﻣﯽﮐﻨﻨﺪ و ﺑﻪ ﺗﺮﺗﯿﺐ ﺑﻪ ﻫﻢ ﻣﺮﺗﺒﻂاﻧﺪ. ﻣﺜﺎل: ﻣﺎژوﻟﯽ ﮐﻪ ﺑﺮای ﭘﺮدازش ﯾﮏ ﻓﺮم، اﺑﺘﺪا دادهﻫﺎ را درﯾﺎﻓﺖ، ﺳﭙﺲ اﻋﺘﺒﺎرﺳﻨﺠﯽ، و در ﻧﻬﺎﯾﺖ ذﺧﯿﺮه

ﻣﯽﮐﻨﺪ.

#### (ارﺗﺒﺎﻃﯽ) Communicational Cohesion .5

اﺟﺰا ﺑﺮ روی دادهﻫﺎی ﻣﺸﺘﺮک ﮐﺎر ﻣﯽﮐﻨﻨﺪ و ﺑﻪ ﻫﻤﯿﻦ دﻟﯿﻞ ﺑﻪ ﻫﻢ ﻣﺮﺗﺒﻂ ﻫﺴﺘﻨﺪ. ﻣﺜﺎل: ﻣﺎژوﻟﯽ ﮐﻪ ﻫﻢ دادهﻫﺎی ﯾﮏ ﺟﺪول دﯾﺘﺎﺑﯿﺲ را ﺑﻪروزرﺳﺎﻧﯽ ﻣﯽﮐﻨﺪ و ﻫﻢ ﮔﺰارشﻫﺎﯾﯽ از آن ﺗﻬﯿﻪ

ﻣﯽﮐﻨﺪ.

#### (ﺗﺮﺗﯿﺒﯽ) Sequential Cohesion .6

ﺧﺮوﺟﯽ ﯾﮏ ﺟﺰء، ﻣﺴﺘﻘﯿﻤ ًﺎ ورودی ﺟﺰء دﯾﮕﺮ اﺳﺖ و اﯾﻦ ارﺗﺒﺎط ﺗﺮﺗﯿﺐ ﻣﻨﻄﻘﯽ اﯾﺠﺎد ﻣﯽﮐﻨﺪ. ﻣﺜﺎل: ﻣﺎژوﻟﯽ ﮐﻪ اﺑﺘﺪا دادهﻫﺎ را از ﻓﺎﯾﻞ ﻣﯽﺧﻮاﻧﺪ، ﺳﭙﺲ ﭘﺮدازش ﻣﯽﮐﻨﺪ، و در ﻧﻬﺎﯾﺖ ﺧﺮوﺟﯽ را ذﺧﯿﺮه

ﻣﯽﮐﻨﺪ.

#### (ﮐﺎرﮐﺮدی) Functional Cohesion .7

ﺑﺎﻻﺗﺮﯾﻦ ﺳﻄﺢ ﻫﻢﭘﯿﻮﺳﺘﮕﯽ اﺳﺖ؛ ﻣﺎژول ﻓﻘﻂ ﯾﮏ وﻇﯿﻔﻪ ﻣﺸﺨﺺ دارد و ﻫﻤﻪ اﺟﺰا ﺑﺮای اﻧﺠﺎم آن ﮐﺎر ﻣﯽﮐﻨﻨﺪ.

ﻣﺜﺎل: ﯾﮏ ﻣﺎژول ﮐﻪ ﻓﻘﻂ وﻇﯿﻔﻪ ﻣﺤﺎﺳﺒﻪ ﻣﯿﺎﻧﮕﯿﻦ ﯾﮏ ﺳﺮی داده را دارد.

ﻣﺤﺎﺳﺒﻪ :Cohesion

.1 ﺷﻤﺎرش ﺗﻌﺪاد وﻇﺎﯾﻒ ﻣﺮﺗﺒﻂ در ﻣﺎژول .2 ﻣﺤﺎﺳﺒﻪ ﻧﺴﺒﺖ وﻇﺎﯾﻒ ﻣﺮﺗﺒﻂ ﺑﻪ ﮐﻞ وﻇﺎﯾﻒ

.3 درﺻﺪﮔﺬاری (%100-0)

### Coupling )ﺟﻔﺖﺷﺪﮔﯽ(

**ﺟﻔﺖﺷﺪﮔﯽ** ﺑﻪ ﻣﯿﺰان واﺑﺴﺘﮕﯽ ﻣﺎژولﻫﺎ ﺑﻪ ﯾﮑﺪﯾﮕﺮ اﺷﺎره دارد. ﻫﺮﭼﻪ واﺑﺴﺘﮕﯽ ﺑﯿﺸﺘﺮ ﺑﺎﺷﺪ، ﺟﻔﺖﺷﺪﮔﯽ ﺑﺎﻻﺗﺮ اﺳﺖ

ﮐﻪ ﻣﻌﻤﻮ ﻻ ﻣﻄﻠﻮب ﻧﯿﺴﺖ. اﻧﻮاع ﺟﻔﺖﺷﺪﮔﯽ از ﮐﻤﺘﺮﯾﻦ ﺗﺎ ﺑﯿﺸﺘﺮﯾﻦ ﺷﺪت واﺑﺴﺘﮕﯽ ﺑﻪ ﺷﺮح زﯾﺮ اﺳﺖ:

.1 **Coupling Data )دادهای(** ﺳﺎدهﺗﺮﯾﻦ و ﮐﻤﺘﺮﯾﻦ ﺳﻄﺢ ﺟﻔﺖﺷﺪﮔﯽ اﺳﺖ. ﻣﺎژولﻫﺎ ﻓﻘﻂ از ﻃﺮﯾﻖ ﺗﺒﺎدل ﭘﺎراﻣﺘﺮﻫﺎی ﺳﺎده ﻣﺜﻞ اﻋﺪاد ﯾﺎ

رﺷﺘﻪﻫﺎ ﺑﺎ ﯾﮑﺪﯾﮕﺮ ارﺗﺒﺎط دارﻧﺪ.

ﻣﺜﺎل: ﺗﺎﺑﻌﯽ ﮐﻪ ﻣﻘﺪار ﻋﺪدی را ﺑﻪ ﻋﻨﻮان ﭘﺎراﻣﺘﺮ ﻣﯽﮔﯿﺮد و ﻋﻤﻠﯿﺎت اﻧﺠﺎم ﻣﯽدﻫﺪ.

#### (ﻣﻬﺮ ) Stamp Coupling .2

ﻣﺎژولﻫﺎ ﺑﺎ اﺳﺘﻔﺎده از ﺳﺎﺧﺘﺎرﻫﺎی دادهای )ﻣﺜﻞ آﺑﺠﮑﺖﻫﺎ ﯾﺎ آراﯾﻪﻫﺎ( اﻃﻼﻋﺎت را ردوﺑﺪل ﻣﯽﮐﻨﻨﺪ.

ﻣﺜﺎل: ﺗﺎﺑﻌﯽ ﮐﻪ ﯾﮏ آﺑﺠﮑﺖ ﮐﺎرﺑﺮ را ﻣﯽﮔﯿﺮد و اﻃﻼﻋﺎت آن را ﺗﻐﯿﯿﺮ ﻣﯽدﻫﺪ.

.3 **Coupling Control )ﮐﻨﺘﺮﻟﯽ(** ﯾﮑﯽ از ﻣﺎژولﻫﺎ ﺑﺎ اﺳﺘﻔﺎده از ﭘﺎراﻣﺘﺮﻫﺎی ﮐﻨﺘﺮﻟﯽ )ﻣﺜﻞ ﻓﻠﮓﻫﺎ ﯾﺎ ﺷﺮاﯾﻂ( ﺟﺮﯾﺎن ﻋﻤﻠﯿﺎت ﻣﺎژول دﯾﮕﺮ را ﻣﺸﺨﺺ

ﻣﯽﮐﻨﺪ.

ﻣﺜﺎل: ﺗﺎﺑﻌﯽ ﮐﻪ ﯾﮏ ﻓﻠﮓ «true/false» ﻣﯽﮔﯿﺮد و ﺑﺮ اﺳﺎس آن ﯾﮏ ﻋﻤﻠﯿﺎت ﺧﺎص را اﺟﺮا ﻣﯽﮐﻨﺪ.

#### (ﺧﺎرﺟﯽ) External Coupling .4

ﻣﺎژولﻫﺎ ﺑﻪ ﭘﺮوﺗﮑﻞﻫﺎ، ﻓﺮﻣﺖﻫﺎی دادهای، ﯾﺎ ﺳﯿﺴﺘﻢﻫﺎی ﺧﺎرﺟﯽ واﺑﺴﺘﻪ ﻫﺴﺘﻨﺪ.

ﻣﺜﺎل: ﯾﮏ ﻣﺎژول ﮐﻪ ﺑﻪ ﯾﮏ ﺳﺮوﯾﺲ API ﺧﺎرﺟﯽ ﻣﺘﺼﻞ ﻣﯽﺷﻮد.

#### (ﻣﺸﺘﺮک) Common Coupling .5

ﭼﻨﺪ ﻣﺎژول ﺑﻪ ﻣﺘﻐﯿﺮﻫﺎی ﺳﺮاﺳﺮی variables) (global دﺳﺘﺮﺳﯽ دارﻧﺪ. اﯾﻦ ﻧﻮع ﺟﻔﺖﺷﺪﮔﯽ ﺧﻄﺮﻧﺎک اﺳﺖ،

زﯾﺮا ﺗﻐﯿﯿﺮ در اﯾﻦ ﻣﺘﻐﯿﺮﻫﺎ ﺑﺮ ﻫﻤﻪ ﻣﺎژولﻫﺎ اﺛﺮ ﻣﯽﮔﺬارد.

ﻣﺜﺎل: ﻣﺎژولﻫﺎﯾﯽ ﮐﻪ ﻫﻤﻪ از ﯾﮏ ﻣﺘﻐﯿﺮ global ﺑﻪ ﻧﺎم «current\_user» اﺳﺘﻔﺎده ﻣﯽﮐﻨﻨﺪ.

#### (ﻣﺤﺘﻮاﯾﯽ) Content Coupling .6

ﺷﺪﯾﺪﺗﺮﯾﻦ و ﺑﺪﺗﺮﯾﻦ ﻧﻮع ﺟﻔﺖﺷﺪﮔﯽ اﺳﺖ. ﻣﺎژول ﻣﺴﺘﻘﯿﻤ ًﺎ ﺑﻪ ﻣﺤﺘﻮای داﺧﻠﯽ ﻣﺎژول دﯾﮕﺮ دﺳﺘﺮﺳﯽ دارد.

ﻣﺜﺎل: ﯾﮏ ﻣﺎژول ﮐﻪ ﻣﺘﺪﻫﺎی ﺧﺼﻮﺻﯽ ﯾﮏ ﮐﻼس دﯾﮕﺮ را ﻣﺴﺘﻘﯿﻤ ًﺎ ﻓﺮاﺧﻮاﻧﯽ ﻣﯽﮐﻨﺪ.

## ﻣﺤﺎﺳﺒﻪ :Coupling

.1 ﺷﻤﺎرش ارﺗﺒﺎﻃﺎت ﺑﯿﻦ ﻣﺎژولﻫﺎ .2 ﺗﻌﯿﯿﻦ ﻧﻮع و ﺷﺪت ارﺗﺒﺎط .3 اﻣﺘﯿﺎزدﻫﯽ (10-1)